Collision-free walk planning for humanoid robots using numerical optimization

Thomas Moulard, Florent Lamiraux, Pierre-Brice Wieber

Collision-free walk planning for humanoid robots using numerical optimization

Thomas Moulard, Florent Lamiraux
LAAS-CNRS, Université de Toulouse
7, avenue du Colonel Roche
31077 Toulouse cedex 4, France
thomas.moulard@laas.fr

Pierre-Brice Wieber
INRIA Rhône-Alpes, Grenoble, France
pierre-brice.wieber@inria.fr

***Abstract*— This paper addresses the problem of walk planning for a humanoid robot on a flat ground cluttered by obstacles. The algorithm we propose works in two steps. In a first step, a collision-free path for the bounding box of the robot moving in the horizontal plane is planned using classical path planning methods. In a second step, a spline is fit on the path and optimized using numerical optimization tools. The optimization criterion is time, constraints are defined by positive distance to obstacles and bounded velocities of the feet. The method has been tested on a real humanoid robot HRP-2.**

I. INTRODUCTION

Humanoid robots are highly complex systems due to their numerous degrees of freedom. As such, computing whole body movement is time-consuming when using classical probabilistic path planning algorithms such as Rapidly-exploring Random Trees (RRT) or Probabilistic Road Maps (PRM) [7] [3].

Considering the whole body is critical for manipulation or contact point planning and leads to complex algorithms trying to solve a problem in a high dimensional space. However, when focusing on the walking problem alone, the problem is much simpler. Indeed, most of the time the movement is realized on a flat ground which allows to search for a 2d trajectory with 3d constraints. An example of this approach is [16] where path planning for a humanoid robot is simplified by considering bounding boxes.

Even in this precise context, walking to a goal position can usually be done in plenty of different ways. Independently of the path which should be as short as possible, trajectory should take into account hardware constraints such as speeds and acceleration bounds. A specific issue is also choosing between walking forward or sideways. Usually, humans behave as cart-like systems [1] but this can change when the environment is especially cluttered with obstacles or when the goal position is near the starting one. [11] presents a method based on numerical optimization combining holonomic and non-holonomic movements which does not take into account obstacles.

On the opposite, approaches such as [2] allow to walk on uneven terrain, step over obstacles or climb stairs. The algorithm relies on a stack of foot prints and chooses the next best foot print through different heuristics. The generality of the presented algorithm is useful for difficult situations but may be inefficient when the robot is walking on a flat ground due to the growing complexity of the underlying graph used for A* based algorithms.

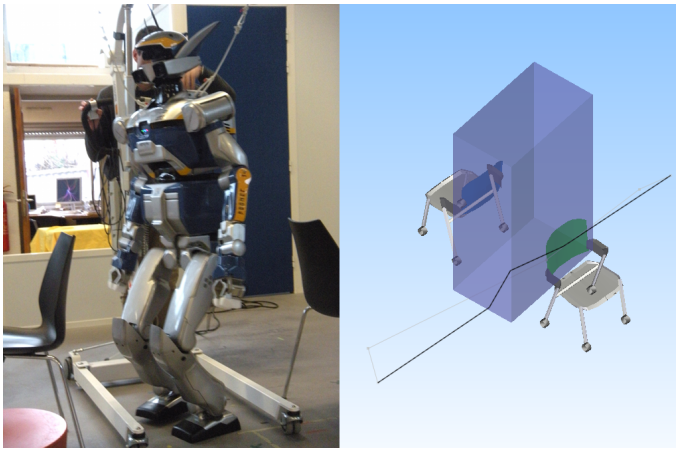

Fig. 1. Humanoid robot HRP-2 passing between two chairs

This paper describes a motion planning method for humanoid robots walking on flat ground with obstacles. The novelty resides in the combination of numerical optimization methods with classical path planning techniques.

II. PLANNING ALGORITHM FOR A HUMANOID ROBOT

To plan a trajectory for a humanoid robot, one efficient method is to first consider the robot as a box moving on a plane. This simplified model contains three degrees of freedom: x, y, θ. Then, a probabilistic path planning algorithm is applied to search for a trajectory. One difference between previous approaches is that this search is not constrained to produce efficient movement. On the opposite, movement between roadmap nodes are classical linear interpolation. When the search tree is not constrained at all and only checks for collision, solving the planning problem on the simplified model is fast. Previous work on optimizing search trees for constraint-free applications includes [4], [15] and [5].

After having computed a rough trajectory for the box, numerical optimization is used to refine the path and transform it into an efficient trajectory. This step will be explained in detail in the next section.

The optimization process done, an efficient and feasible

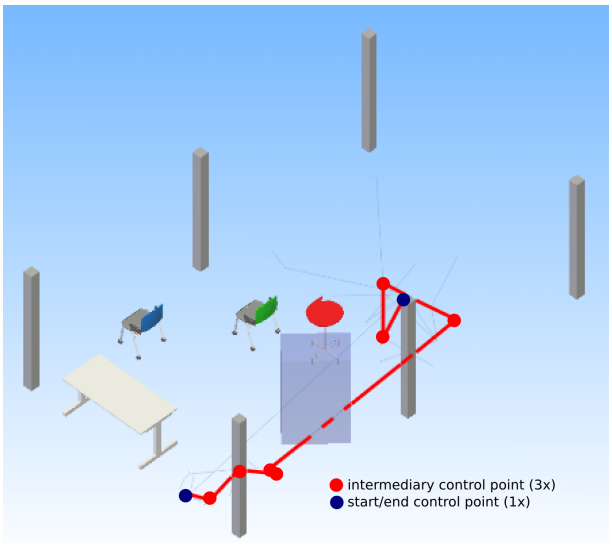

Fig. 2. Fitting a cubic spline on a trajectory

trajectory is obtained. To generate the whole-body movement for the humanoid robot, a stack of foot prints is generated along the box trajectory.

Being given robot foot steps and waist height, it is possible to reconstruct the movement of the lower part of the robot using inverse kinematics [12]. This step also integrates the stability constraint. As a consequence, it may reject some infeasible foot steps. In practice, enforcing a reasonable step length during the foot step generation phase prevents this kind of failure.

III. NUMERICAL OPTIMIZATION FOR PATH PLANNING

Numerical optimization step takes a collision-free trajectory as its input and computes a more efficient trajectory.

Numerical optimization solvers search the vector of parameters which minimizes a cost function under linear and non-linear constraints. The first step is to fit the input trajectory on a parameterized curve. By modifying the curve control points, the solver will be able to change the trajectory as desired.

A. Spline fitting

The proposed algorithm uses cubic splines [13] to fit the trajectory. One interesting spline feature in this case is that if a control point is repeated up to the spline order, the spline will pass through this control point. As the input trajectory generated by the probabilistic method is made of several linear paths (linear interpolation is used to link roadmap nodes), the final trajectory is a set of segments, it makes the fitting process trivial.

To fit the trajectory, one control point is added at the beginning and the end of the trajectory and three control points are added between each segment. To increase the possible movements (Figure 2).

Time	control point 1			...	control point n		
λ	x_0	y_0	θ_0	...	x_n	y_n	θ_n

Fig. 3. Optimization parameters

B. Time scaling

Figure 3 illustrates that an additional parameter λ is added to model the trajectory speed. $\lambda = 1$ means normal speed and $\lambda = 2$ corresponds to a trajectory twice *slower*. The normal speed is defined by the probabilistic method step but is not affected by hardware constraints. At the beginning of the solving step, a λ satisfying all constraints is computed to make sure that the optimization process starts from a valid point.

Let γ be the initial trajectory defined from t_{min} to t_{max}. A free time trajectory Γ is defined from T_{min} to $T_{max} = T_{min} + \lambda(t_{max} - t_{min})$ as

$$\Gamma_\lambda(T) = \gamma(t_{min} + \frac{1}{\lambda}(T - T_{min})) \quad (1)$$

The use of a free time trajectory allows the solver to both optimize trajectory shape and speed by respectively changing the control points and the scale. The free time trajectory derived from the initial trajectory γ is used as the state of the solver.

C. Cost function

The remaining task is to define the optimization problem properly. An optimization problem is defined by the cost function which evaluates the "efficiency" of a trajectory and the constraints which enforces that the solving algorithm will not return an invalid trajectory.

The only factor required in the cost function is time: the trajectory should be as fast as possible while respecting the speed constraints described in the next section.

$$Cost(x) = \lambda \quad (2)$$

D. Speed constraint

Three types of constraints are considered in the problem: constraints on the initial and goal points, speeds and obstacles constraints.

The first constraints indicates that the first and the last control points cannot be moved.

Speeds constraints are computed on each foot separately. The constraint combines frontal and orthogonal speed into one value to penalize more orthogonal speed compared to frontal speed. It means that the robot will be allowed to go much faster when walking forward. As a consequence, as the solver tries to accelerate the trajectory as much as possible, it will also change the control points to prefer forward movements. The constraints is expressed for each foot through the following formula:

$$Speed_{(t,foot)}(x) = (\frac{v^x_{foot}}{v^x_{max}})^2 + (\frac{v^y_{foot}}{v^y_{max}})^2 - 1 \quad (3)$$

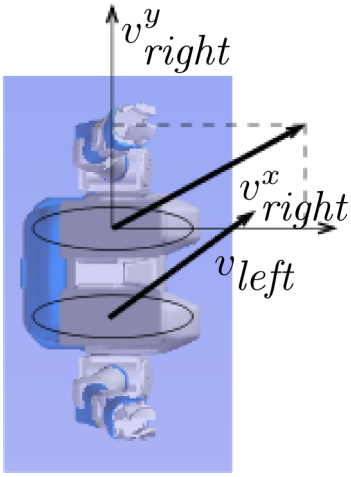

Fig. 4. Frontal and orthogonal speeds computation for each foot

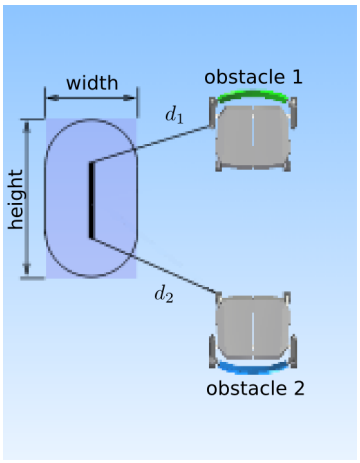

Fig. 5. Distance computations

As illustrated in Figure 4, this constraint defines an ellipse to constrain more lateral speeds. v^x_{foot} is the frontal speed of the specified foot and v^y_{foot} is the orthogonal speed defined as follow:

$$\begin{aligned} v^x_{foot}(q,\dot{q}) = & \begin{pmatrix} cos\ \theta \\ sin\ \theta \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \\ = & \ cos\ \theta\ \dot{x} + sin\ \theta\ \dot{y} \end{aligned} \quad (4)$$

$$\begin{aligned} v^y_{foot}(q,\dot{q}) = & \begin{pmatrix} -sin\ \theta \\ cos\ \theta \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \\ = & -sin\ \theta\ \dot{x} + cos\ \theta\ \dot{y} \end{aligned} \quad (5)$$

In the previous formula, (x, y, θ) represents the foot position in the global frame. q and $\dot{q}$ are respectively the robot configuration and its derivative with respect to time.

This formulation presents the interest of gathering orthogonal speed and frontal speed in one constraint. When constraining separately speeds, high orthogonal speeds trajectories do not cost more than others. On the opposite, by summing speeds, having high orthogonal speeds limits the frontal speed and by consequence the trajectory will have a higher cost.

This constraint is applied along the whole trajectory:

$$\begin{aligned} \forall T \in [T_{min}, T_{max}], foot \in (left, right), \\ Speed_{(T,foot)}(x) \leq 0 \end{aligned} \quad (6)$$

v^x_{max} and v^y_{max} are respectively maximum frontal and orthogonal velocities and depend on the robot.

In practice, a time discretization step is used to add the constraint uniformly along the trajectory.

E. Obstacle constraint

Obstacle constraint is more difficult to express: distance function has to be defined carefully to produce acceptable gradients for the solver. Indeed, the naive approach which associates zero distance to any configuration where the robot and the obstacle are in collision will not work. The reason is that anytime the constraint is not fulfilled, a null gradient will be returned, preventing the solver from knowing in which direction the search should be pursued.

To avoid that pitfall, the distance is computed from the plane inside the bounding box to the obstacle and the minimum distance is increased by half of the box width as illustrated in Figure 5.

The constraint can be expressed as the following formula:

$$Obstacle_{(T,obstacle)}(x) = distance(obstacle, \Gamma(T)) \quad (7)$$

As the speed constraint, the obstacle constraint is discretized along the trajectory:

$$\forall T \in [T_{min}, T_{max}], Obstacle_{(T,obstacle)}(x) \leq d_{min} \quad (8)$$

d_{min} is defined as half of the box width plus a safety distance which depends on the application.

To quickly compute this constraint gradient during the solving process, the method explained in [9] is used. Gradient computation details can be found in the appendix. Compared to numeric differentiation, using this algorithm makes the whole solving process about 100 times faster.

IV. EXPERIMENTAL RESULTS

This approach has been tested using a HRP-2 humanoid platform [6]. This 30 degrees of freedom is 1.54 [m] tall and weights 58 [kg].

The first environment that has been tested is composed of two chairs which are narrow enough to forbid straight forward movement. Bounds on the environment have been set to make sure the RRT has to pass between the chairs to solve the problem. This example illustrates how holonomic and non-holonomic movements are combined in this approach. The second environment (tested in simulation only) is more general: it represents a large room with some obstacles. This illustrates that this approach is still valid when planning a longer trajectory. Computation time for these two examples are detailed in Figure 6

Scenarii	Path planning	Optimization	*Whole process*
Passing between two chairs	8.58s	47.22s	*2 min 05.55 s*
Large environment with obstacles	4.14 s	1 min 9.35 s	*4 min 5.11 s*

Fig. 6. Computation time. Whole process column incorporates probabilistic path planning step, optimization and whole-body movement computation.

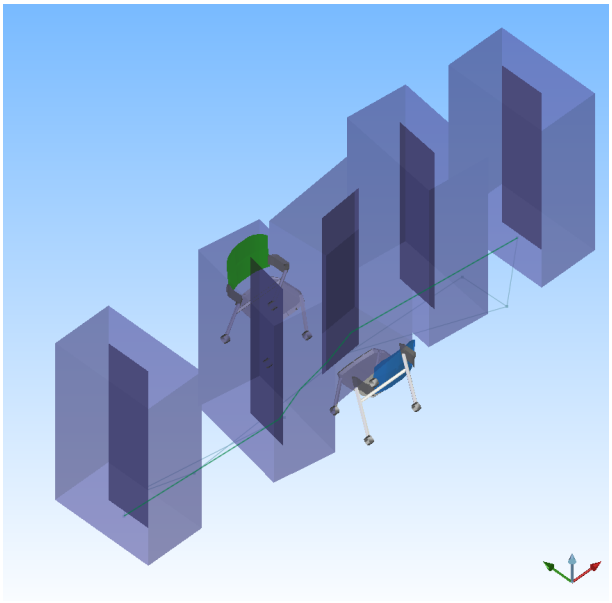

Fig. 7. Passing between two-chairs simulation

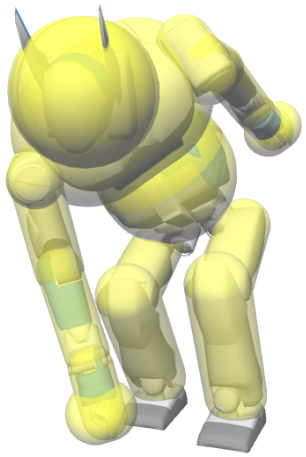

Fig. 8. Simplified model of the robot for posture optimization

To compute these examples, the RobOptim optimization framework[1] and the CFSQP non-linear solver [8] have been used. The KineoWorks SDK[2] has also been used. In particular, the collision detector KCD has played an important role in this approach efficiency.

The reshaped trajectory which has been tested on HRP-2 is shown in Figure 7. This figure illustrates differents states of the bounding box along the trajectory.

The optimization process has been profiled using Valgrind and reveals that most of the time is spent computing constraints and the associated gradient. With the example requiring the robot to pass between two chairs, about 19% of the time was constraints computation and 53% was gradients computation. Constraints incorporating geometry are particularly costly: on the whole computation time, 34% has been spent in the 3d distance computation routines.

V. FUTURE WORK: OPTIMIZING WHOLE BODY TRAJECTORIES

The method presented in the previous section is used to generate trajectories but can be extended to whole body trajectory optimization. Few works such as [10] and [14] are currently exploring this idea. Future work will allow the current framework to optimize the whole robot posture through the same idea: first using search tree to find a rough estimation and then refining it through numerical optimization. Fast computation can be achieved by generalizing the obstacle constraint presented in the previous section to the whole humanoid robot as illustrated by Figure 8. This work may allow to generate better trajectories in difficult cases as numerical optimization can easily saturate a constraint unlike random search trees which usually do not find "limit configurations". However, acceptable computing time and good convergence despite the presence of local minima still have to be demonstrated.

[1] `http://roboptim.sourceforge.net/`
[2] `http://www.kineocam.com/`

VI. CONCLUSION

In this paper, we presented an approach to refine trajectories generated by a probabilistic planning algorithm. Results both in simulation and with real hardware demonstrated that this method generates efficient trajectories. Previous section also illustrated that computation time are acceptable even with complex environment containing multiple obstacles.

However, this method still presents some drawbacks: numerical optimization as any local method will not be able to change globally the structure of a trajectory, it will be limited to local optimization to refine the trajectory. Local minima can also prevent the method from converging toward a better solution. Another problem is that if this method refines properly difficult cases, it is much slower than usual heuristics such as shortcuts. To allow fast optimization when the initial trajectory is extremely inefficient despite a simple environment, the shortcut algorithm is run as a pretreatment to optimize roughly before using numerical optimization.

VII. ACKNOWLEDGMENTS

This work has been supported by the R-BLINK project, ANR-08-JCJC-0075-01. We are very grateful to Oussama Kanoun for providing us with the picture of Figure 8.

APPENDIX

[9] introduces a fast method to compute obstacles distance. This section describes the algorithm proposed in this article.

Let $O(q)$ and $R(q)$ respectively the closed point on the obstacle and the robot when the robot is in configuration q. d represents the distance between these two points:

$$d(q) = ||O(q) - R(q)|| \tag{9}$$

The gradient associated to this formula is:

$$\frac{\partial d(q)}{\partial q} = \frac{(O(q) - R(q))^T}{||O(q) - R(q)||}(\frac{\partial O}{\partial q}(q) - \frac{\partial R}{\partial q}(q)) \tag{10}$$

In the previous equation, most of the computation time is due to the two derivatives. As closest points vary with the configuration, the equation seems at first sight costly and would greatly slow the optimization process.

Fortunately, a more efficient formulation is possible by considering a reference frame attached to the considered body: $(B(q), e_1(q), \ldots, e_d(q))$. The point considered on the robot can be expressed as:

$$R(q) = \sum_{l=1}^{d} \rho_l(q) e_l(q) \tag{11}$$

In the previous equation, $(\rho_1(q), \ldots, \rho_l(q))$ are local coordinates of $R(q)$. The gradient can then be expressed as:

$$\frac{\partial R}{\partial q}(q) = \sum_{l=1}^{d} \frac{\partial \rho_l}{\partial q}(q) e_l(q) + \sum_{l=1}^{d} \rho_l(q) \frac{\partial e_l(q)}{\partial q} \tag{12}$$

This formula can be explained through the law of compositions of motions. The two parts of the sum are respectively:

- the relative motion of $R(q)$,
- the absolute motion of the point coinciding with $R(q)$.

As the relative motion of $R(q)$ stays in the body boundary, it is by consequence orthogonal to vector $O(q) - R(q)$:

$$(O(q) - R(q))^T \sum_{l=1}^{d} \frac{\partial \rho_l}{\partial q}(q) e_l(q) = 0 \tag{13}$$

Following the same idea with $\frac{\partial O}{\partial q}(q)$, we can conclude that the relative motion on the obstacle is orthogonal to vector $O(q) - R(q)$:

$$(O(q) - R(q))^T \frac{\partial O}{\partial q}(q) = 0 \tag{14}$$

The gradient finally be expressed as:

$$\frac{\partial d}{\partial q}(q) = \frac{(R(q) - O(q))^T}{||O(q) - R(q)||} \frac{\partial R_{\in body}}{\partial q} \tag{15}$$

$\frac{\partial R_{\in body}}{\partial q} = \sum_{l=1}^{d} \rho_l(q) \frac{\partial e_l(q)}{\partial q}$ and represents the absolute velocity induced by variations of q, of the point of the body coinciding with $R(q)$.

REFERENCES

[1] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. Optimizing principles underlying the shape of trajectories in goal oriented locomotion for humans. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 131–136, Dec. 2006.

[2] Joel Chestnutt, James Kuffner, Koichi Nishiwaki, and Satoshi Kagami. Planning biped navigation strategies in complex environments. In *in IEEE Int. Conf. Humanoid Robots*, 2003.

[3] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, June 2005.

[4] E. Ferre and J.-P. Laumond. An iterative diffusion algorithm for part disassembly. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 3149–3154 Vol.3, April-1 May 2004.

[5] L. Jaillet, A. Yershova, S.M. La Valle, and T. Simeon. Adaptive tuning of the sampling domain for dynamic-domain rrts. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2851–2856, Aug. 2005.

[6] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1083–1090 Vol.2, 26-May 1, 2004.

[7] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006.

[8] Craig Lawrence, Jian L. Zhou, and Andr L. Tits. User's guide for cfsqp version 2.5: A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical report, 1997.

[9] O. Lefebvre, F. Lamiraux, and D. Bonnafous. Fast computation of robot-obstacle interactions in nonholonomic trajectory deformation. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4612–4617, April 2005.

[10] S. Miossec, K. Yokoi, and A. Kheddar. Development of a software for motion optimization of robots - application to the kick motion of the hrp-2 robot. pages 299 –304, Dec. 2006.

[11] K. Mombaur, J.-P. Laumond, and E. Yoshida. An optimal control model unifying holonomic and nonholonomic walking. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 646–653, Dec. 2008.

[12] O. Stasse, B. Verrelst, P.-B. Wieber, B. Vanderborghtand P. Evrard, A. Kheddar, and K. Yokoi. Modular architecture for humanoid walking pattern prototyping and experiments. *Advanced Robotics, Special Issue on Middleware for Robotics –Software and Hardware Module in Robotics System*, 22(6):589–611, 2008.

[13] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, New York, 3 edition, August 2002.

[14] W. Suleiman, E. Yoshida, J.-P. Laumond, and A. Monin. On humanoid motion optimization. pages 180 –187, 29 2007-Dec. 1 2007.

[15] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *Robotics, IEEE Transactions on*, 23(1):151–157, Feb. 2007.

[16] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond. Humanoid motion planning for dynamic tasks. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 1–6, Dec. 2005.